RECEIVED
CENTRAL FAX CENTER

MAY 18 2006

BEYER WEAVER & THOMAS, LLP

INTELLECTUAL PROPERTY LAW

590 W. El Camino Real, Mountain View, CA 94040
Telephone: (650) 961-8300 Facsimile: (650) 961-8301
www.beyerlaw.com

FACSIMILE COVER SHEET

May 18, 2006

Receiver: U.S. Patent and Trademark Office

TEL #:

FAX # : (571) 273-8300

Sender: Susan W. Xu for Ramin Mahboubian

Our Ref. No.: SUN1P275

Re: Application No. 09/641,035

Pages Including Cover Sheet(s): 07

MESSAGE:

Sir:

Please file the attached Applicant Initiated Interview Request for the above referenced application.

RECEIVED
CENTRAL FAX CENTER
MAY 18 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: David Wallman	Attorney Docket No.: SUN1P275/P4783
Application No.: 09/641,035	Examiner: KANG, Insun
Filed: August 16, 2000	Group: 2193
Title: METHOD AND APPARATUS FOR CACHING NATIVE CODE IN A VIRTUAL MACHINE INTERPRETER	Confirmation No.: 3756

CERTIFICATE OF FACSIMILE TRANSMISSION
I hereby certify that this correspondence is being transmitted by facsimile to fax number 571-273-8300 of the U.S. Patent and Trademark Office on May 18, 2006.

Signed: _____
Susan W. Xu

APPLICANT INITIATED INTERVIEW REQUEST FORM

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Tentative Participants:

1) R. Mahboubian
2)
3)
4)

Proposed Date of Interview: **May 31, 2006** Proposed Time: **2:00 PM (Eastern Time)**

Type of Interview Requested:

☒ Telephone ☐ Personal ☐ Video Conference

Exhibit to be Shown or Demonstrated: ☐ Yes ☒ No
If yes, provide brief description:

ISSUES TO BE DISCUSSED

Issues (Rej., Obj., etc.)	Claims/ Fig., #s	Prior Art	Discussed	Agreed	Not Agreed
1) 102	Claim 1	*Lethin et al.*	☐	☐	☐
2) 101	Calim 28		☐	☐	☐
3)			☐	☐	☐

BRIEF DESCRIPTION OF AGRUMENTS TO BE PRESENTED:

a) *Lethin et al.* does NOT teach or suggest copying native code generated by an interpreter into a cache

It is noted U.S. Patent Application Publication No. 2002/0147969 (*Lethin et al.*) states that:

> If the correct branch is not in the cache, i.e. "no" in step S404, then flow proceeds to step S408 and one BRANCH_L1_RECORD (i.e. the record containing all fields which may be updated, such as encountered_sub_count and taken_sub_count) in the set designated by "S" above is removed from the L1 cache and written to the branch log. Next, the current branch information is written into the set designated by "S". Moreover, during writing of the current branch record into the set "S", the current branch record is placed as the first element of the set. This is because the same branch will very likely be executed again, thereby increasing performance and efficiency of the system. In other words sets S404 will be executed faster. Even when the branch is in the cache, i.e. "yes", it may be copied to the branch log if it has been executed a large number of times since it was last flushed. (Paragraph 168 of *Lethin et al.*)

However, it is respectfully submitted that step 408 described by *Lethin et al.* does NOT teach or suggest copying native code generated by an interpreter into a cache. This distinction is evident as Fig. 11 of *Lethin et al.* "illustrates a branch logging method" (Paragraph 157) that "counts how many times a branch has executed" (Paragraph 146).

Moreover, it is respectfully submitted that *Lethin et al.* does NOT teach or suggest this feature. Instead, *Lethin et al.* pertains to a "translation" system for "translation of target object code in the field of object code translators" when "it becomes necessary to convert object code which has been developed for one computer on another computer having a different computer architecture" (Paragraph 2). In other words, the interpreter of *Lethin et al.* is "for individually translating object code into corresponding translated object code" (paragraph 11). As such, *Lethin et al.* states that interpreter can be further used for: "determining a number of executions of branch instructions in the source code, and a compiler [can be used] for grouping instructions of source object code into a segment when a number of executions of a corresponding branch instruction exceeds a threshold number, and for dynamically compiling the segment (Paragraph 11). Clearly, it is in the context of this translation of object code (not execution of virtual machine instructions) that the branch logging method depicted in Fig. 11 of used to "count how many times a branch has been executed" (Paragraph

146). However, it is respectfully submitted that *Lethin et al.* does NOT teach or suggest copying native code that has been effectively generated by an interpreter after execution of a virtual machine program instruction.

b) *Lethin et al.* does NOT teach or suggest: determining by an interpreter that executes virtual machine program instructions whether a basic block that includes native code corresponding to the virtual machine program instruction to be executed by the interpreter

Contrary to the Examiner's assertion, it is respectfully submitted that neither the interpreter of *Lethin et al.* executes virtual machine program instructions, nor does the L1 cache of *Lethin et al.* store native code corresponding to the virtual machine instruction. Again, the interpreter of *Lethin et al.* translates object code but it is not used for execution of virtual machine instructions. Furthermore, the cache of *Lethin et al.* is used to store counts of how many times a branch has been executed (Paragraph 146), but it is not used to store native code corresponding to a virtual machine program instruction.

c) *Lethin et al.* does NOT teach or suggest executing native code corresponding to a virtual machine program instruction from a cache

In view of the foregoing, it is respectfully submitted that *Lethin et al.* does NOT teach or suggest this additional feature.

1. (Currently Amended) A computer-implemented method for executing computer code by ~~increasing the performance of~~ a virtual machine that uses an interpreter to execute virtual machine program instructions, the computer-implemented method comprising:

obtaining, by an interpreter, a virtual machine program instruction to be executed by the virtual machine;

determining, by the interpreter, whether the virtual machine program instruction is a branch instruction;

determining, by the interpreter, whether a basic block is present in a code cache that stores native code corresponding to virtual machine program instructions when it is determined that the virtual machine program instruction is a branch instruction to be executed by said virtual machine, wherein the basic block is associated with a case block of the interpreter, wherein the case block corresponds to ~~associated with~~ the virtual machine program instruction and the basic block would include ~~includes~~ native code corresponding to the virtual machine program instruction ~~that has been previously interpreted and executed by the interpreter~~, thereby determining by the interpreter whether native code needed to execute the virtual machine program instructions is available in the code cache ~~being associated with the virtual machine~~;

executing~~, by the interpreter,~~ the native code included in the basic block present in ~~from said~~ the code cache when it is determined by the interpreter that the basic block is present in the code cache and the virtual machine program instruction is a branch instruction;

interpreting, by the interpreter, the virtual machine instruction when said determining determines that the basic block associated with the case block of the interpreter is not present in ~~said~~ the code cache, wherein said interpreting generates native code for the virtual machine instruction; and

copying the native code generated by the interpreter into ~~said cache~~ the code cache after the interpreting of the virtual machine program instruction ~~the code~~ by the interpreter when said determining determines that the basic block associated with the case block of the interpreter is not present in ~~said~~ the code cache so that subsequently the native code can be obtained from the code cache, thereby enhancing the performance of the virtual machine.

28. (Currently Amended) A computer system for increasing the performance of a virtual machine that uses an interpreter to execute virtual machine program instructions, wherein the computer system is operable ~~capable of operating~~ to:

obtain a virtual machine program instruction to be executed by the virtual machine;

determine whether the virtual machine program instruction is a branch instruction;

determine whether a basic block is present in a code cache that stores native code corresponding to virtual machine program instructions when it is determined that the virtual machine program instruction is a branch instruction to be executed by said virtual machine, wherein the basic block is associated with a case block of the interpreter, wherein the case block corresponds to ~~associated with~~ the virtual machine program instruction and the basic block would include ~~includes~~ native code corresponding to the virtual machine program instruction ~~that has been previously interpreted and executed by the interpreter~~, thereby determining by the interpreter whether native code needed to execute the virtual machine program instructions is available in the code cache;

execute the native code included in the basic block present in ~~from said~~ the code cache when it is determined by the interpreter that the basic block is present in the code cache and the virtual machine program instruction is a branch instruction;

interpret the virtual machine instruction when said determining determines that the basic block associated with the case block of the interpreter is not present in ~~said~~ the code cache, wherein said interpreting generates native code for the virtual machine instruction; and

copy the native code generated by the interpreter into the ~~cache~~ code cache after the interpreting of the virtual machine program instruction ~~the code~~ by the interpreter when said determining determines that the basic block associated with the case block of the interpreter is not present in ~~said~~ the code cache so that subsequently the native code can be obtained from the code cache, thereby enhancing the performance of the virtual machine.

An interview was conducted on the above-identified application on .

*Note: This form should be completed by applicant and submitted to the examiner in advance of the interview (see MPEP §713.01). This application will not be delay from issue because of applicant's failure to submit a written record of this interview. Therefore, applicant is advised to file a statement of the substance of this interview (37 CFT 1.33(b)) as soon as possible.

(Applicant/Applicant's Representative) Signature)

(Examiner/SPE Signature)